mushroom!
(Classifying Mushrooms as Edible or Poisonous)

arya bharath and james wright

Background/Goal

- Real world implications
  - Most people will not be able to identify whether something is definitively poisonous or not based on looking at it alone
    - What is the optimal method for classifying?
    - Possible to create a very simple set of rules that humans can understand and use
      - Would lose a lot of precision that would result in worse results over time
  - Useful to use a machine learning model to classify instead
- Classify mushrooms poisonous or edible
- 20 attributes

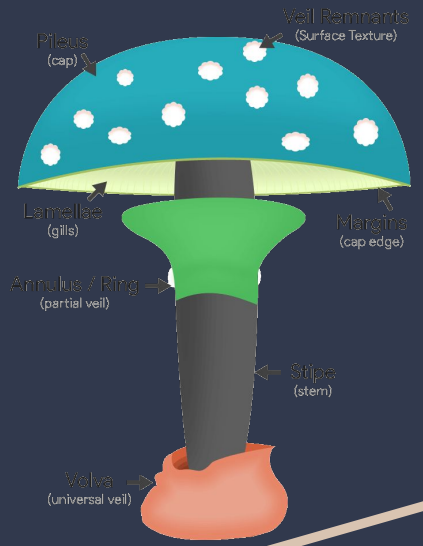
Dataset

secondary_data

class	cap-diameter	cap-shape	cap-surface	cap-color	does-bruise-or-bleed	gill-attachment	gill-spacing	gill-color	stem-height	stem-width	stem-root	stem-surface	stem-color	veil-type	veil-color	has-ring	ring-type	spore-print-color	habitat	season
p	15.26	x	g	o	f	e		w	16.95	17.09	s	y	w	u	w	t	g		d	w
p	16.6	x	g	o	f	e		w	17.99	18.19	s	y	w	u	w	t	g		d	u
p	14.07	x	g	o	f	e		w	17.8	17.74	s	y	w	u	w	t	g		d	w
p	14.17	f	h	e	f	e		w	15.77	15.98	s	y	w	u	w	t	p		d	w
p	14.64	x	h	o	f	e		w	16.53	17.2	s	y	w	u	w	t	p		d	w
p	15.34	x	g	o	f	e		w	17.84	18.79	s	y	w	u	w	t	p		d	u
p	14.85	f	h	o	f	e		w	17.71	16.89	s	y	w	u	w	t	g		d	w
p	14.86	x	h	e	f	e		w	17.03	17.44	s	y	w	u	w	t	p		d	u
p	12.85	f	g	o	f	e		w	17.27	18.69	s	y	w	u	w	t	p		d	a
p	13.55	f	g	e	f	e		w	16.04	16.88	s	y	w	u	w	t	p		d	w
p	14.17	f	h	e	f	e		w	17.86	18.02	s	y	w	u	w	t	p		d	a
p	13.4	x	h	o	f	e		w	17.95	17.14	s	y	w	u	w	t	p		d	u
p	17.37	x	h	o	f	e		w	18.1	18.27	s	y	w	u	w	t	g		d	u
p	16.56	x	h	o	f	e		w	18.89	18.11	s	y	w	u	w	t	p		d	u
p	15.37	x	h	e	f	e		w	18.19	18.38	s	y	w	u	w	t	g		d	a
p	15.54	f	h	e	f	e		w	18.26	17.87	s	y	w	u	w	t	p		d	a
p	15.19	x	g	e	f	e		w	17.42	17.67	s	y	w	u	w	t	p		d	a
p	17.4	x	h	o	f	e		w	18.48	18.54	s	y	w	u	w	t	p		d	a
p	16.16	x	g	o	f	e		w	19.46	18.9	s	y	w	u	w	t	p		d	w
p	16.93	x	h	e	f	e		w	18.31	17.81	s	y	w	u	w	t	p		d	u
p	13.0	f	g	e	f	e		w	17.3	17.19	s	y	w	u	w	t	g		d	w
p	13.06	x	g	e	f	e		w	16.9	17.38	s	y	w	u	w	t	g		d	u
p	17.23	x	h	o	f	e		w	17.63	17.92	s	y	w	u	w	t	g		d	w
p	14.39	x	g	e	f	e		w	16.98	17.48	s	y	w	u	w	t	p		d	w
p	15.56	f	g	o	f	e		w	17.28	16.99	s	y	w	u	w	t	p		d	u
p	13.2	x	g	o	f	e		w	16.91	16.9	s	y	w	u	w	t	p		d	w
p	13.9	f	h	e	f	e		w	17.81	17.77	s	y	w	u	w	t	g		d	u
p	16.38	f	g	o	f	e		w	18.36	17.99	s	y	w	u	w	t	p		d	u
p	13.3	x	h	e	f	e		w	17.6	17.52	s	y	w	u	w	t	p		d	a
p	15.95	f	g	o	f	e		w	16.42	16.65	s	y	w	u	w	t	g		d	w
p	16.58	x	g	o	f	e		w	18.26	18.12	s	y	w	u	w	t	g		d	a
p	14.1	f	g	o	f	e		w	16.94	16.76	s	y	w	u	w	t	p		d	u
p	13.81	f	h	o	f	e		w	17.22	17.11	s	y	w	u	w	t	g		d	a
p	14.67	f	h	o	f	e		w	17.2	16.77	s	y	w	u	w	t	p		d	a
p	14.23	f	h	e	f	e		w	17.22	17.23	s	y	w	u	w	t	p		d	u
p	14.78	f	g	e	f	e		w	17.44	17.13	s	y	w	u	w	t	p		d	u
p	15.08	x	g	e	f	e		w	17.09	17.8	s	y	w	u	w	t	g		d	w
p	16.0	x	g	e	f	e		w	16.96	17.22	s	y	w	u	w	t	g		d	u
p	14.76	f	g	o	f	e		w	18.14	17.54	s	y	w	u	w	t	p		d	a
p	14.89	x	g	e	f	e		w	17.78	18.07	s	y	w	u	w	t	g		d	w
p	13.05	f	g	e	f	e		w	17.32	17.17	s	y	w	u	w	t	p		d	w
p	13.03	f	h	e	f	e		w	17.05	17.07	s	y	w	u	w	t	p		d	w
p	14.07	f	g	o	f	e		w	16.98	17.76	s	y	w	u	w	t	p		d	w
p	15.09	f	g	o	f	e		w	17.45	18.15	s	y	w	u	w	t	g		d	a
p	12.22	f	h	o	f	e		w	18.19	17.62	s	y	w	u	w	t	g		d	w
p	15.12	x	g	e	f	e		w	17.03	17.2	s	y	w	u	w	t	g		d	a
p	18.04	f	g	e	f	e		w	18.86	19.79	s	y	w	u	w	t	g		d	w
p	12.27	f	g	o	f	e		w	16.15	17.17	s	y	w	u	w	t	g		d	w
p	16.03	f	g	o	f	e		w	19.34	19.37	s	y	w	u	w	t	g		d	u
p	16.51	x	g	o	f	e		w	18.38	18.29	s	y	w	u	w	t	g		d	w

- UCI Repository
- 61,069 instances
- Dimensionality of 20
- 307,463 missing values total
  - Concentrated in only a few specific columns
  - How to deal with missing values?
    - If we replace with mean/mode, it is possible that our filled values make results worse (maybe only certain types of mushrooms had values for certain attributes, which could skew our results)
    - Also possible to use WEKA placeholder missing values
    - Is no information better than potentially bad information?
- Class distribution
  - 33,888 poisonous (55.5%)
  - 27,181 edible (44.5%)
  - Unbalanced dataset

Attributes

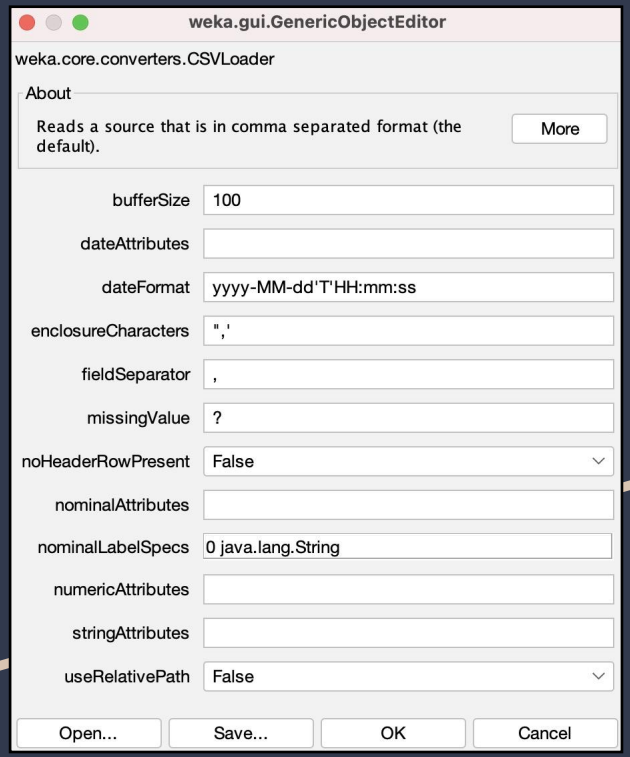


1. **cap-diameter**: quantitative continuous, given as a float in centimeters (cm)
2. **cap-shape**: qualitative nominal, given as bell (b), conical (c), convex (x), flat (f), sunken (s), spherical (p), or others (o)
3. **cap-surface**: qualitative nominal, given as fibrous (i), grooves (g), scaly (y), smooth (s), shiny (h), leathery (l), silky (k), sticky (t), wrinkled (w), or fleshy (e)
4. **cap-color**: qualitative nominal, given as brown (n), buff (b), gray (g), green (r), pink (p), purple (u), red (e), white (w), yellow (y), blue (l), orange (o), or black (k)
5. **does-bruise-bleed**: qualitative nominal, given as bruises-or-bleeding (t) or no (f)
6. **gill-attachment**: qualitative nominal, given as adnate (a), adnexed (x), decurrent (d), free (e), sinuate (s), pores (p), none (f), or unknown (?)
7. **gill-spacing**: quantitative continuous, given as a float in centimeters (cm)
8. **gill-color**: qualitative nominal, given as brown (n), buff (b), gray (g), green (r), pink (p), purple (u), red (e), white (w), yellow (y), blue (l), orange (o), black (k), or none (f)
9. **stem-height**: quantitative continuous, given as a float in centimeters (cm)
10. **stem-width**: quantitative continuous, given as a float in centimeters (cm)

Attributes continued

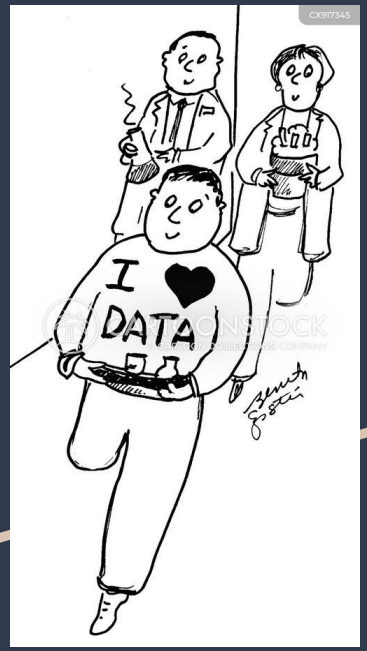
1. **stem-root**: qualitative nominal, given as bulbous (b), swollen (s), club (c), cup (u), equal (e), rhizomorphs (z), or rooted (r)
2. **stem-surface**: qualitative nominal, given as fibrous (i), grooves (g), scaly (y), smooth (s), shiny (h), leathery (l), silky (k), sticky (t), wrinkled (w), fleshy (e), or none (f)
3. **stem-color**: qualitative nominal, given as brown (n), buff (b), gray (g), green (r), pink (p), purple (u), red (e), white (w), yellow (y), blue (l), orange (o), black (k), or none (f)
4. **veil-type**: qualitative nominal, given as partial (p) or universal (u)
5. **veil-color**: qualitative nominal, given as brown (n), buff (b), gray (g), green (r), pink (p), purple (u), red (e), white (w), yellow (y), blue (l), orange (o), black (k), or none (f)
6. **has-ring**: qualitative nominal, given as ring (t) or none (f)
7. **ring-type**: qualitative nominal, given as cobwebby (c), evanescent (e), flaring (r), grooved (g), large (l), pendant (p), sheathing (s), zone (z), scaly (y), movable (m), none (f), or unknown (?)
8. **spore-print-color**: qualitative nominal, given as brown (n), buff (b), gray (g), green (r), pink (p), purple (u), red (e), white (w), yellow (y), blue (l), orange (o), or black (k)
9. **habitat**: qualitative nominal, given as grasses (g), leaves (l), meadows (m), paths (p), heaths (h), urban (u), waste (w), or woods (d)
10. **season**: qualitative nominal, given as spring (s), summer (u), autumn (a), or winter (w)

Preprocessing



- 2 approaches for missing values
  - Replaced with “?” using WEKA
    - Default missingValue for WEKA, we wanted to explore how this interacted
  - Mean/mode
- Fixed issue with dataset not loading in WEKA with GenericObjectEditor window
  - WEKA assumes .csv files have values separated by commas, but our values had semicolons
  - Edited *fieldSeparator* value
- Class variable was originally set to the last attribute in the file (**habitat**)
- No normalization necessary
  - All quantitative attributes (those with width, height, etc.) were relatively small and were confined to the same ranges of values
- No derived attributes

Train/Validation/Test Split



- To split our dataset into training, validation, and testing sets, we used a Python script that employed pandas, sklearn, scipy.io, and the arff libraries
  - Used the stratify parameter to ensure that the training, validation, and testing datasets would all maintain the same proportions between class labels
  - For our data, the ratio between p (poisonous) to e (edible) instances was around 1.25, so all of the split datasets would also need to have this ratio
  - 70:15:15 split between the training, validation, and testing datasets
    - 42,672 instances, 9174 instances, and 9175 instances respectively

Python Script

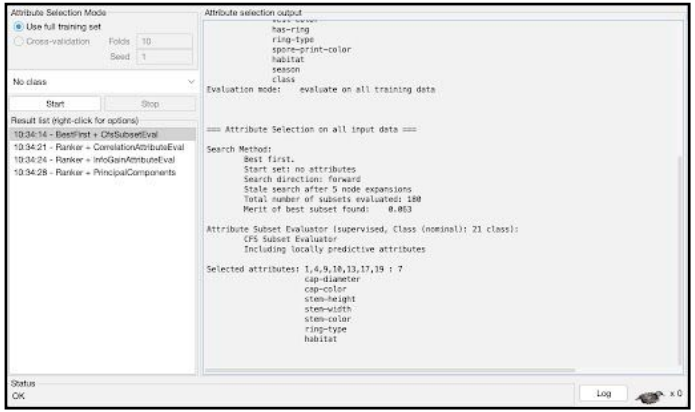
```
import pandas as pd
from sklearn.model_selection import train_test_split
from scipy.io import arff
import arff as liac_arff

data, meta = arff.loadarff('data.arff')
df = pd.DataFrame(data)

def decode_bytes(df):
    for col in df.select_dtypes([object]):
        df[col] = df[col].apply(lambda x: x.decode('utf-8')
if isinstance(x, bytes) else x)
    return df

df = decode_bytes(df)

X = df.iloc[:, :-1]
y = df.iloc[:, -1]

X_train, X_temp, y_train, y_temp = train_test_split(X, y,
test_size=0.3, stratify=y, random_state=42)

X_val, X_test, y_val, y_test = train_test_split(X_temp,
y_temp, test_size=0.5, stratify=y_temp, random_state=42)

train_set = pd.concat([y_train, X_train], axis=1)
val_set = pd.concat([y_val, X_val], axis=1)
test_set = pd.concat([y_test, X_test], axis=1)

def save_to_arff(df, filename, meta):
    arff_data = {
        'description': '',
        'relation': 'dataset',
        'attributes': [(name, list(df[name].unique())) if
df[name].dtype == 'object' else (name, 'REAL') for name in
df.columns],
        'data': df.values.tolist()
    }
    with open(filename, 'w') as f:
        liac_arff.dump(arff_data, f)

save_to_arff(train_set, 'train_set.arff', meta)
save_to_arff(val_set, 'val_set.arff', meta)
save_to_arff(test_set, 'test_set.arff', meta)
```

Attribute Selection

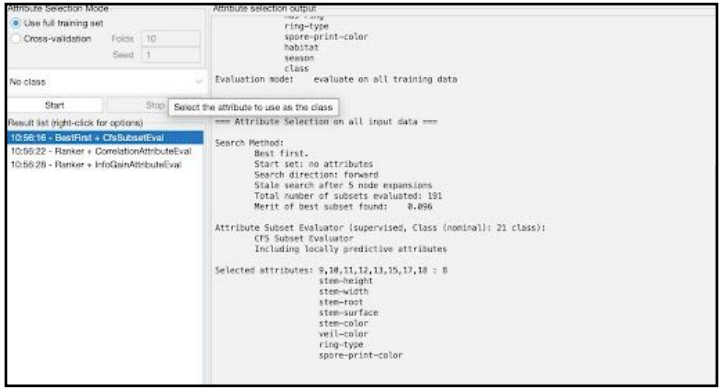
- CfsSubsetEval
  - Uses a combination of the individual predicting power of each attribute and the redundancy of the predictions to select attributes
- CorrelationAttributeEval
  - Orders attributes by using Pearson Correlation between each individual attribute and the class
- InfoGainAttributeEval
  - Sorts attributes based on information gain/entropy loss
- OneRAttributeEval
  - Chooses attributes with OneR Classifier (uses a ruleset based on one attribute)
- Intuitive selection

CfsSubsetEval

- Missing values as “?”

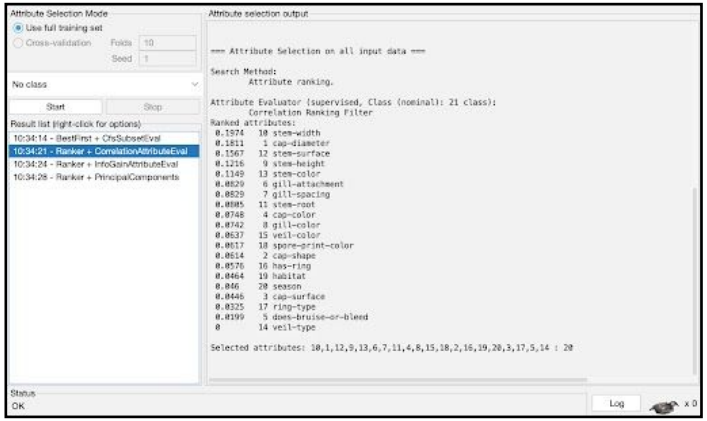


- Missing values as mean/mode

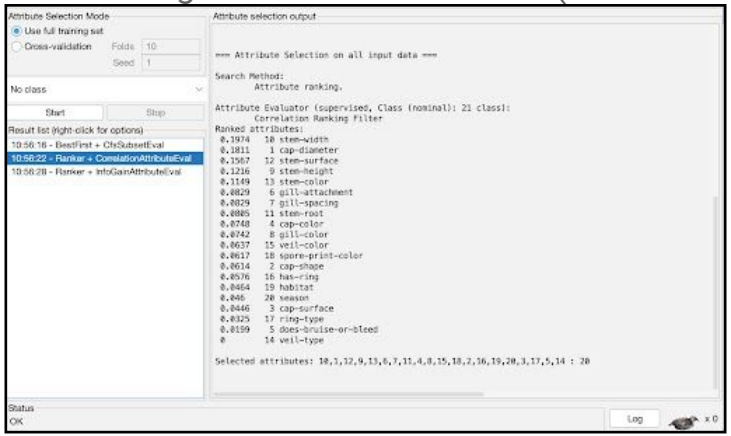


CorrelationAttributeEval

- Missing values as “?” (cut-off 0.08)

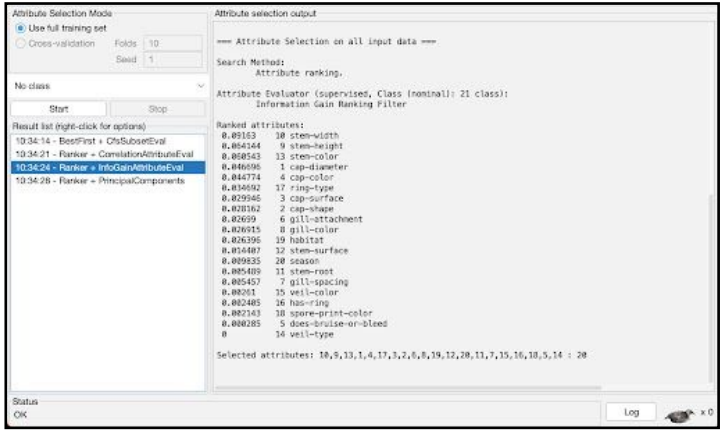


- Missing values as mean/mode (cut-off 0.08)

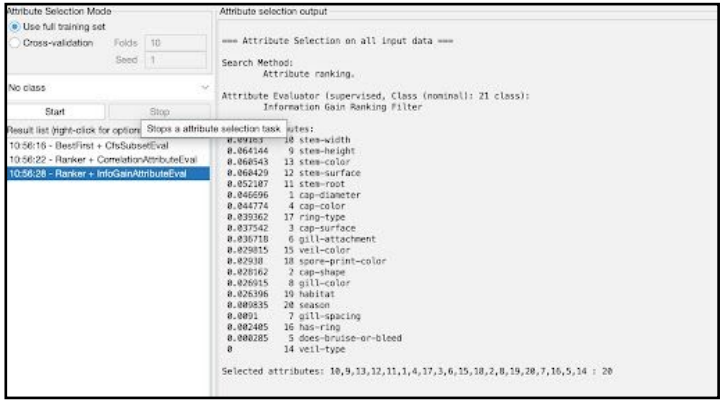


InfoGainAttributeEval

- Missing values as “?” (cut-off 0.025)

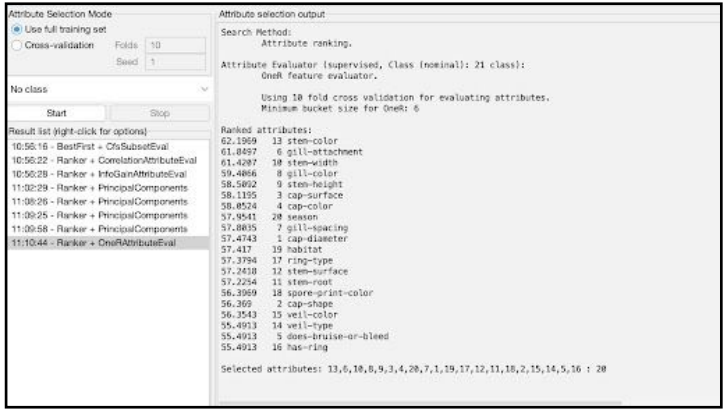


- Missing values as mean/mode (cut-off 0.04)

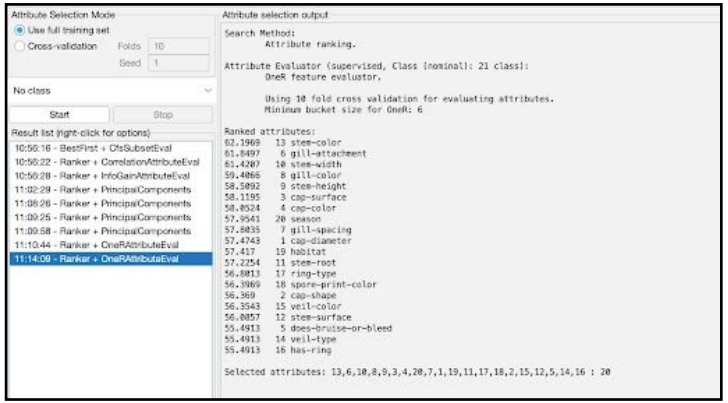


OneRAttributeEval

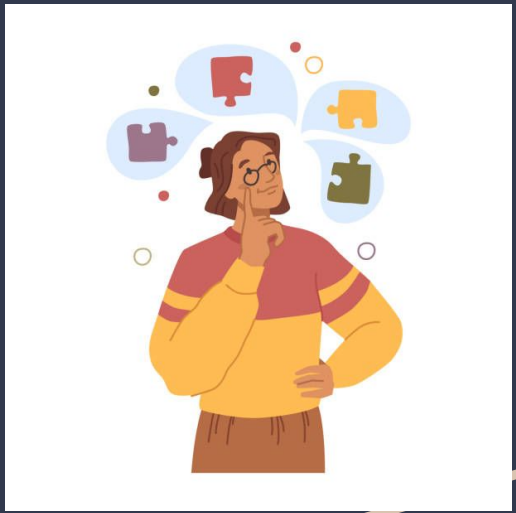
- Missing values as “?” (cut-off 57.5)



- Missing values as mean/mode (cut-off 57.5)



Intuitive Selection



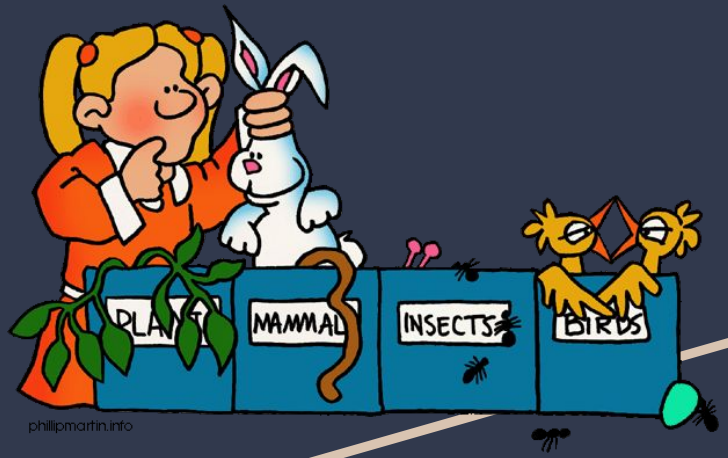
Cap-shape
Cap-surface
Cap-color
Gill-color
Stem-surface
Stem-color
Veil-type
Veil-color
Has-ring
Ring-type
Spore-print-color

- We chose these because poisonous plants and animals are usually identifiable by their shape and color, not necessarily their size
    - We thus removed the attributes that had to do with the measurements of the mushrooms, as well as some others like season and habitat that we thought were likely not related as much to the class

Common Selected Attributes

- Several attributes consistently appeared in the attribute selection algorithms
  - Stem-width
  - Stem-height
  - Stem-color
  - Cap-color
  - Gill-attachment
  - Cap-diameter
- We did our intuitive selection based on what made the most logical sense to us, but maybe the performance of the intuitive models would be better if these attributes were chosen instead
  - Can also run additional attribute selection algorithms on WEKA to more confidently identify important attributes to predicting class

Classifiers



- NaiveBayes
- DecisionTable
- RandomForest
- J48

NaiveBayes

- Probabilistic classifier based on Bayes' Theorem
  - Assumes attributes are independent of each other
- Average accuracy lower
  - Maybe is lower because the attributes are not actually completely independent
  - Discussed in later slides

Naive Bayes	Accuracy	TP Rate	FP Rate	ROC Area	PRC Area
CfsSubsetNoMissing	71.5	0.715	0.293	0.771	0.768
CfsSubsetEval	61.6	0.617	0.431	0.716	0.715
CorrelationNoMissing	66.5	0.665	0.383	0.77	0.76
Correlation	63.1	0.631	0.417	0.752	0.745
InfoGainNoMissing	64.1	0.641	0.407	0.749	0.745
InfoGain	67.7	0.676	0.359	0.769	0.762
OneRNoMissing	67.7	0.676	0.345	0.761	0.752
OneR	68.1	0.681	0.34	0.759	0.749
IntuitionNoMissing	74.6	0.746	0.252	0.817	0.817
Intuition	73.3	0.733	0.268	0.812	0.815
average accuracy NB	67.82				

DecisionTable

- ML classifier using a table based on attributes/conditions
- When new data point introduced, algorithm evaluates attributes against conditions specified in decision table
- Identifies matching rules based on evaluated conditions

DecisionTable	Accuracy	TP Rate	FP Rate	ROC Area	PRC Area
CfsSubsetNoMissing	87.6	0.876	0.116	0.956	0.955
CfsSubsetEval	89.8	0.898	0.103	0.965	0.964
CorrelationNoMissing	93.8	0.938	0.007	0.988	0.987
Correlation	97.2	0.972	0.031	0.997	0.996
InfoGainNoMissing	91	0.91	0.094	0.971	0.97
InfoGain	97.7	0.977	0.024	0.997	0.997
OneRNoMissing	97.7	0.977	0.026	0.997	0.996
OneR	98.4	0.984	0.018	0.999	0.999
IntuitionNoMissing	96.7	0.967	0.038	0.994	0.994
Intuition	96.7	0.967	0.035	0.996	0.996
average accuracy DT	94.66				

J48

- Decision tree algorithm, recursively splits data into branches based on attribute values
  - Splits tree based on which attribute has the highest information gain (change in entropy)
  - Can deal with missing values
  - Builds tree and then uses for classification

J48	Accuracy	TP Rate	FP Rate	ROC Area	PRC Area
CfsSubsetNoMissing	93.3	0.933	0.065	0.986	0.984
CfsSubsetEval	96.5	0.965	0.035	0.994	0.994
CorrelationNoMissing	98.8	0.988	0.012	0.998	0.998
Correlation	80	0.8	0.175	0.871	0.865
InfoGainNoMissing	97.6	0.976	0.026	0.997	0.997
InfoGain	99.2	0.992	0.009	0.999	0.999
OneRNoMissing	99.5	0.995	0.006	0.999	0.999
OneR	98.5	0.985	0.014	0.996	0.996
IntuitionNoMissing	97.7	0.977	0.024	0.998	0.998
Intuition	88	0.88	0.113	0.954	0.953
average accuracy J48	94.91				

RandomForest

- Combines many decision trees to produce a best result
  - Splits up data into random subsets
  - Uses each subset to produce a decision tree
  - Because this is a classification problem, each tree will produce a decision
  - Decisions are counted as votes, the classification with most votes is chosen as the output of the classifier
- Many perfect results - why?

RandomForest	Accuracy	TP Rate	FP Rate	ROC Area	PRC Area
CfsSubsetNoMissing	100	1	0	1	1
CfsSubsetEval	100	1	0	1	1
CorrelationNoMissing	100	1	0	1	1
Correlation	99.9	1	0	1	1
InfoGainNoMissing	100	1	0	1	1
InfoGain	100	1	0	1	1
OneRNoMissing	100	1	0	1	1
OneR	99.9	1	0	1	1
IntuitionNoMissing	98	0.98	0.021	0.999	0.999
Intuition	97.7	0.977	0.023	0.998	0.998
average accuracy RF	99.55				

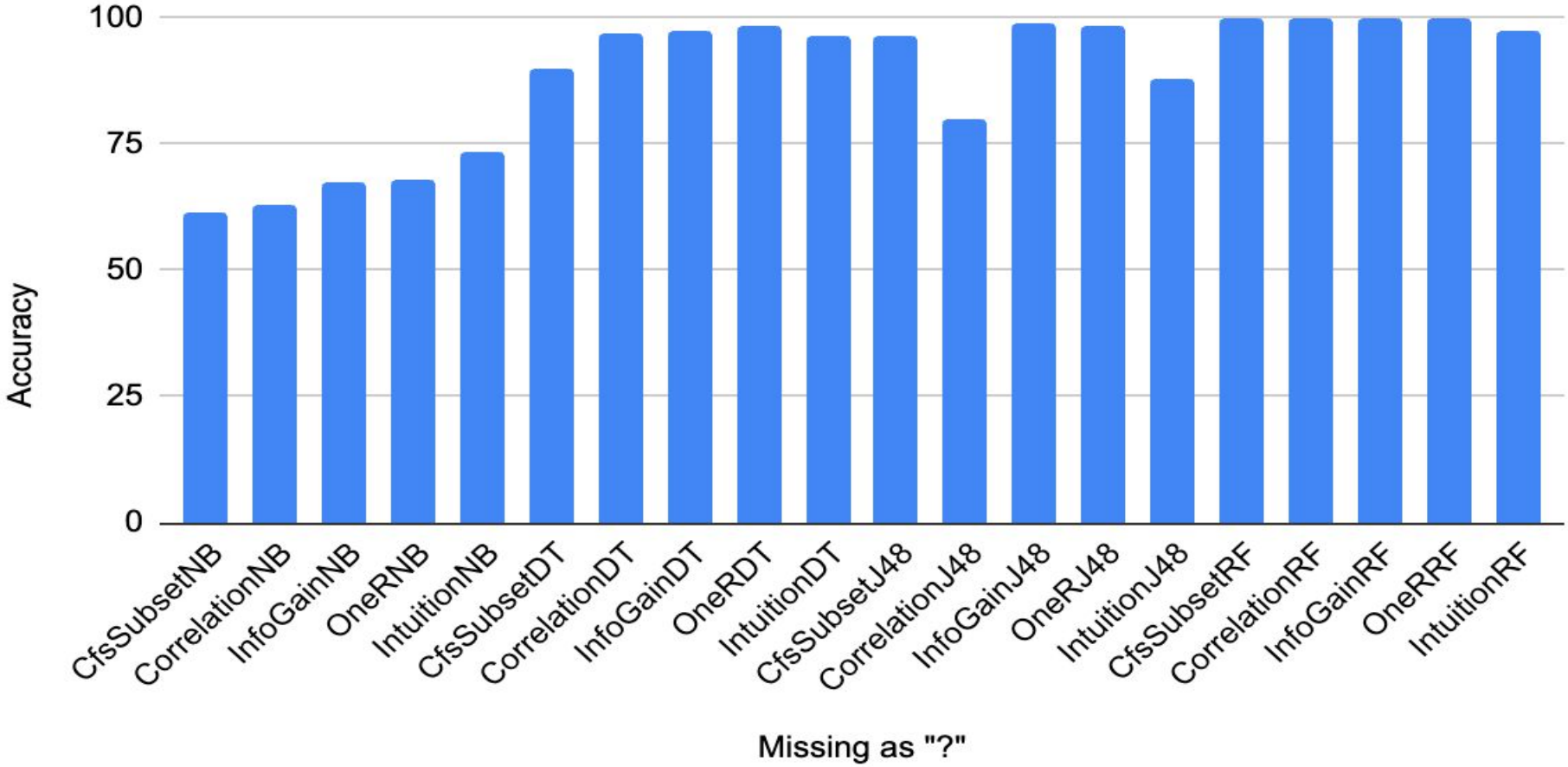
Missing as “?”

Missing as "?"	Accuracy	TP Rate	FP Rate	ROC Area	PRC Area
CfsSubsetNB	61.6	0.617	0.431	0.716	0.715
CorrelationNB	63.1	0.631	0.417	0.752	0.745
InfoGainNB	67.7	0.676	0.359	0.769	0.762
OneRNB	68.1	0.681	0.34	0.759	0.749
IntuitionNB	73.3	0.733	0.268	0.812	0.815
CfsSubsetDT	89.8	0.898	0.103	0.965	0.964
CorrelationDT	97.2	0.972	0.031	0.997	0.996
InfoGainDT	97.7	0.977	0.024	0.997	0.997
OneRDT	98.4	0.984	0.018	0.999	0.999
IntuitionDT	96.7	0.967	0.035	0.996	0.996
CfsSubsetJ48	96.5	0.965	0.035	0.994	0.994
CorrelationJ48	80	0.8	0.175	0.871	0.865
InfoGainJ48	99.2	0.992	0.009	0.999	0.999
OneRJ48	98.5	0.985	0.014	0.996	0.996
IntuitionJ48	88	0.88	0.113	0.954	0.953
CfsSubsetRF	100	1	0	1	1
CorrelationRF	99.9	1	0	1	1
InfoGainRF	100	1	0	1	1
OneRRF	99.9	1	0	1	1
IntuitionRF	97.7	0.977	0.023	0.998	0.998
average accura	88.665				

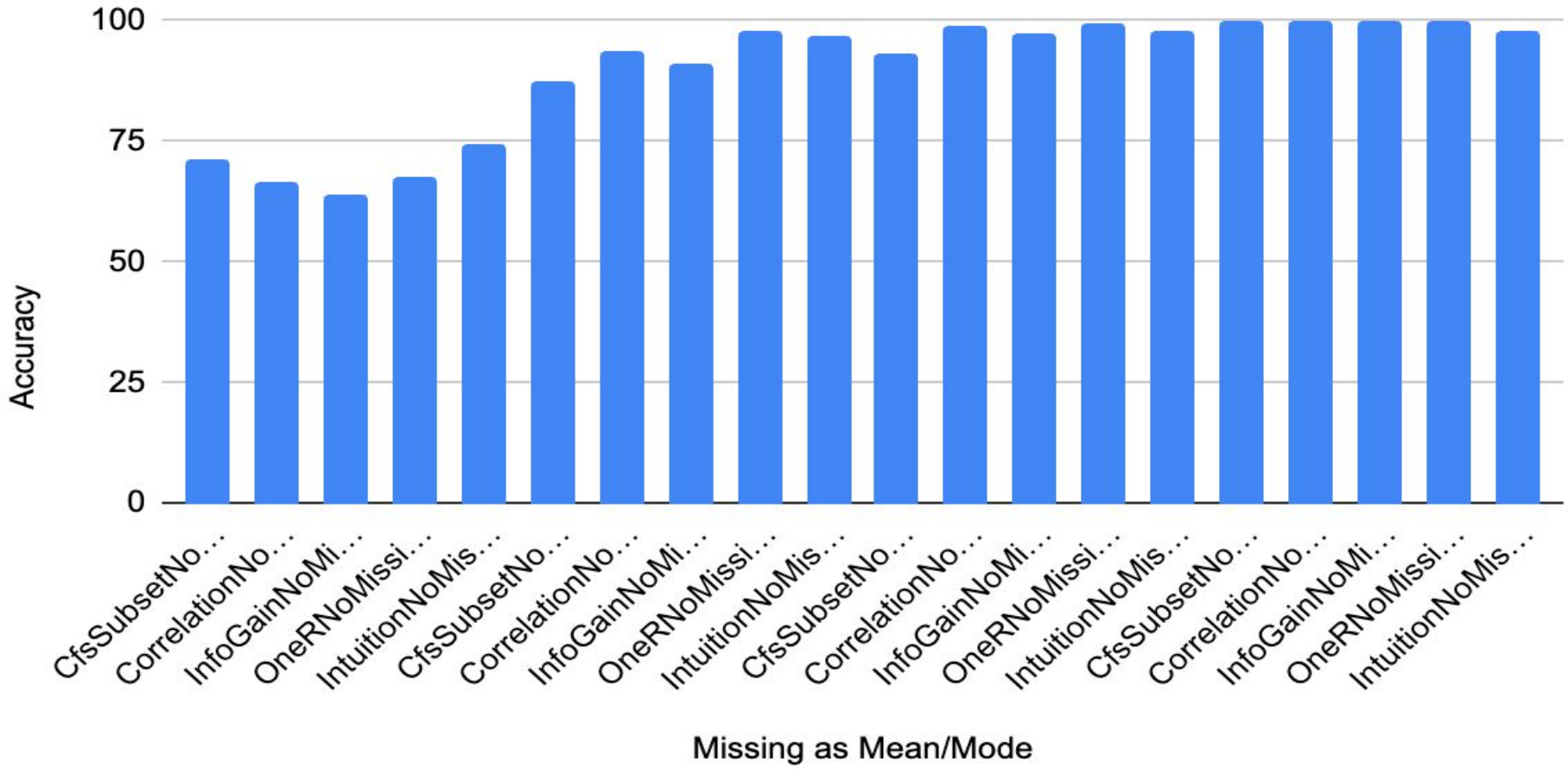
Missing as mean/mode

Missing as Mea	Accuracy	TP Rate	FP Rate	ROC Area	PRC Area
CfsSubsetNoMis	71.5	0.715	0.293	0.771	0.768
CorrelationNoMi	66.5	0.665	0.383	0.77	0.76
InfoGainNoMissi	64.1	0.641	0.407	0.749	0.745
OneRNoMissing	67.7	0.676	0.345	0.761	0.752
IntuitionNoMissir	74.6	0.746	0.252	0.817	0.817
CfsSubsetNoMis	87.6	0.876	0.116	0.956	0.955
CorrelationNoMi	93.8	0.938	0.007	0.988	0.987
InfoGainNoMissi	91	0.91	0.094	0.971	0.97
OneRNoMissing	97.7	0.977	0.026	0.997	0.996
IntuitionNoMissir	96.7	0.967	0.038	0.994	0.994
CfsSubsetNoMis	93.3	0.933	0.065	0.986	0.984
CorrelationNoMi	98.8	0.988	0.012	0.998	0.998
InfoGainNoMissi	97.6	0.976	0.026	0.997	0.997
OneRNoMissing	99.5	0.995	0.006	0.999	0.999
IntuitionNoMissir	97.7	0.977	0.024	0.998	0.998
CfsSubsetNoMis	100	1	0	1	1
CorrelationNoMi	100	1	0	1	1
InfoGainNoMissi	100	1	0	1	1
OneRNoMissing	100	1	0	1	1
IntuitionNoMissir	98	0.98	0.021	0.999	0.999
average accura	89.805				

Accuracy vs. Missing as "?"



Accuracy vs. Missing as Mean/Mode



Best Model

- 6 models with perfect accuracy, so how to choose?
  - Other metrics - error loss
  - 0.0211 root-mean-square error for best model
- randomForest with OneR attribute selection
  - Missing values as mean/mode

```
Time taken to build model: 0.35 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.12 seconds

=== Summary ===

Correctly Classified Instances        9161              100      %
Incorrectly Classified Instances         0                0      %
Kappa statistic                          1
Mean absolute error                      0.0062
Root mean squared error                  0.0211
Relative absolute error                  1.2615 %
Root relative squared error              4.2381 %
Total Number of Instances             9161

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     p
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     e
Weighted Avg.    1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000

=== Confusion Matrix ===

    a    b   <-- classified as
 5084    0 |    a = p
    0 4077 |    b = e
```

NaiveBayes performance

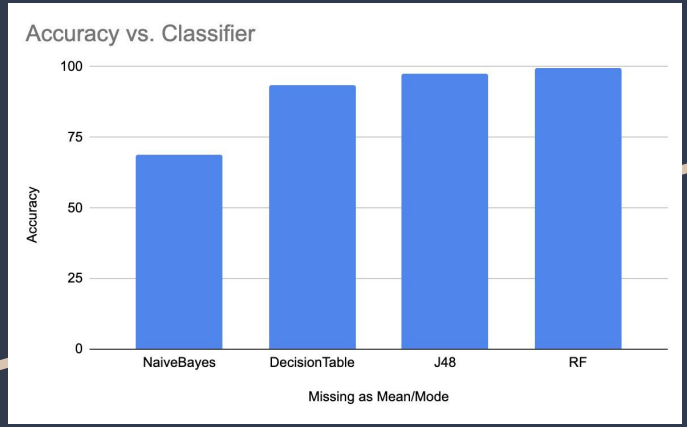
- NaiveBayes assumes all conditions are independent
- For our datasets, clearly several attributes are dependent on each other
  - Quantitative attributes such as **cap-diameter**, **stem-width**, **stem-height** could be related
    - Bigger mushrooms have larger values for all 3?
  - Color-based attributes (**stem-color**, **cap-color**, **veil-color**, **gill-color**) potentially related
  - More general attributes like **habitat** and **season** are so broad that they likely correlate with many of the attributes
- How to improve accuracy?
  - With this condition in mind, maybe it is better to evaluate the relationship between attributes to decrease correlation instead of relating them to the class variable
  - Could this improve model strength?
- In contrast, all other 3 algorithms are based on decision tree algorithms, which can identify relationships between attributes
  - J48, DecisionTable, RandomForest all performed very well
  - Could potentially indicate that many attributes are related

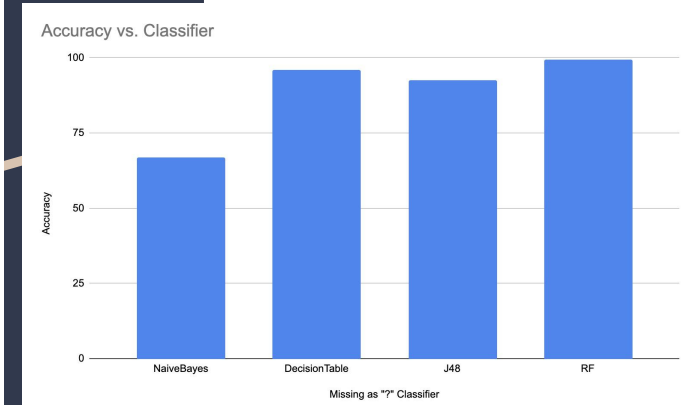
RandomForest performance

- 6 of the models we tested with RandomForest had a perfect accuracy
- With 9,000+ instances in the test set, is this really possible?
  - RandomForest combines predictions of many decision trees
  - If this accuracy was on the training data, this would be a case of possible overfitting - but this is test set
    - What if general variations across entire dataset are small enough RandomForest algorithm can “learn” the testing data as it trains?
  - Also possible that class variable values (poisonous and edible) require such different values for predictions that the line between the two is obvious to a complex model like RandomForest

Missing values - which approach was better?

- Missing values as “?”
  - 88.67% average accuracy
  - Performed noticeably better for DecisionTable
    - Why?
- Missing values as mean/mode
  - 89.81% average accuracy
  - Performed noticeably better for J48
    - Why?
- Interactions with WEKA algorithms and set missing values, but not sure





"?"	Accuracy
NaiveBayes	66.76
DecisionTable	95.96
J48	92.44
RF	99.5

Mean/Mode	Accuracy
NaiveBayes	68.88
DecisionTable	93.36
J48	97.38
RF	99.6

Discussion

- We learned how to use WEKA to successfully train and test a model that classified our own selected data of poisonous vs edible mushrooms with 100% accuracy
- Our dataset initially started off with 20 attributes, and our model could have possibly been improved had we not removed some of the attributes during our attribute selection step
- Even with many missing values, using mean/mode was still better than assigning a specific value for missing items
- Many of the attributes that were removed could have held solid predictive power of whether or not a certain mushroom is edible or poisonous
    - Better attribute selection could be a key improvement from our project
- NaiveBayes and RandomForest performances

Sources

https://archive.ics.uci.edu/dataset/848/secondary+mushroom+dataset